Refine Search

Search Results -

Terms	Documents
L2 and scan	11

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L6

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Monday, April 25, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L6	L2 and scan	11	L6
L5	L2 same (gasket or interface)	13	L5
L4	l2 same fabric	1	L4
L3	L2 same test$	8	L3
L2	L1 same fpga	65	L2
L1	embedded near3 (core or device)	10523	L1

END OF SEARCH HISTORY

L6: Entry 1 of 11 File: USPT Dec 7, 2004

DOCUMENT-IDENTIFIER: US 6829751 B1
TITLE: Diagnostic architecture using FPGA core in system on a chip design

Detailed Description Text (3):
The present invention implements an FPGA core as an embedded Field Programmable Gate Array core that may be used to enable logic to be programmed after the silicon has been produced. The FPGA core can be used to implement on-chip diagnostics to enable debugging functions, such as bus monitoring, probing, single step running, triggering, capturing, etc. One example of such an FPGA may be found in co-pending application Ser. No. 09/464,741, which is hereby incorporated by reference in its entirety.

Detailed Description Text (9):
The system 100 may provide a chip diagnostics architecture by implementing the FPGA core 116. By using the FPGA core 116 to implement such chip diagnostics, simultaneous probing of internal signals can be achieved while the system 100 is running under predetermined conditions (e.g., a normal mode of operation). Additionally, a process may be implemented to allow the FPGA core 116 to collect data from the registers 120a-120n and 122a-122n using a scan chain, while the system 100 is running under predetermined conditions (e.g., a step mode configuration) controlled by the FPGA core 116.

Detailed Description Text (12):
By controlling the system clock, the FPGA core 116 may also be programed to run the system 100 in a single step mode. By utilizing the scan chain under the single step mode, the FPGA core 116 may collect data from the register blocks 110 and 114 implemented on the system 100. Since the scan chain is usually very long (e.g., several thousands of registers per scan chain), the scan chain may be separated into small segments to speed up data collection.

Detailed Description Text (13):
Referring to FIG. 3, a diagram illustrating the breaking of a scan chain into segments is shown. The scan chain implementation of the circuit 102 generally comprises a multiplexer logic block (or circuit) 160 and a scan chain segment block (or circuit) 162. The multiplexer logic block 160 generally comprises a number of multiplexers 164a-164n. The scan chain segment block 162 generally comprises a number of multiplexers 166a-166n and a number of scan chain segments (e.g., SCAN_SEG0, SCAN_SEG1 . . . SCAN_SEGX) . The scan chain segments may allow the FPGA core 116 select and collect register data more quickly.

Detailed Description Text (15):
The following steps outline collecting of register data using the segmented scan chain under single step mode:

Detailed Description Text (18):
(iii) the FPGA core 116 generates the clock signal SYS_CLK, the clock signal DIAG_CLK and select signals DIAG_SEL10 and DIAG_SEL11 (which may be multi-bit or single bit signals) to control which scan segments need to be accessed;

Detailed Description Text (19):

(iv) under the single step mode, after each effective edge of the system clock SYS_CLK, the data in the selected scan segments SCAN_SEG0-SCAN_SEGX may be shifted to the FPGA core 116, then shifted back to the selected scan segment SCAN_SEG0-SCAN_SEGX, and then the clock signal SYS_CLK may resume; and

Detailed Description Text (28):
Referring to FIG. 6, a work flow (or system) 300 of an example software is shown. The software will read in the probe name file, in which all the signals that can be probed are logged. The netlist file and the RTL codes of the design are read. All the registers on different scan segments are mapped to the netlist and related to the RTL codes. In this way, the user can easily decide which signals need to be observed during the debugging period. While the chip 102 is running at normal speed, the software can display any of the internal signals that are connected to the I/O of the FPGA core 116. While the chip 102 is running at single step mode, the software can display all the signals on the chip 102. This is because under single step mode, the FPGA core 116 can be programmed to access any of the on-chip registers. All the combinational signals can be derived from the related register values and the netlist information. By using such software, users can also program the FPGA core 116 to implement different debugging functions. The debugging workstation 104 and the CAD software can be reused in different projects, resulting in reduction of cost.

Detailed Description Text (47):
The FPGA core 116 may simultaneously probe multiple internal signals. By utilizing the scan chain under the single step mode and with the on-chip FPGA core 116 acting as the data process center, all the signals on the chip 102 can be observed. The FPGA core 116 can be used to bridge the signals between different modules, and the under test mode, to isolate a specific module and drive signals to test the specific module. The FPGA core 116 can also be used to add or verify bug fixes. The process of the debugging workstation 104 working with the on-chip FPGA core 116 to generate many powerful debugging features may also be implemented.

CLAIMS:

18. The system according to claim 15, wherein a scan chain is used to diagnose or fix a bug via the logic portion.

Previous Doc Next Doc Go to Doc#

L6: Entry 4 of 11 File: USPT Aug 13, 2002

DOCUMENT-IDENTIFIER: US 6434735 B1
TITLE: Method for programming an FPGA and implementing an FPGA interconnect

Abstract Text (1):
An apparatus comprising a plurality of register logic circuits, a core circuit, a memory circuit, and a plurality of logic circuits. The register logic circuits may each be configured to generate a first logic signal in response to (i) an input data signal, (ii) a second logic signal, (iii) a first clock signal and (iv) a second clock signal. The core circuit may be configured to generate a plurality of data signals and a first control signal in response to the first logic signals and a second control signal. The memory may be configured to present the second control signal to the core circuit. The logic circuits may each be configured to present the second logic signal in response to the first logic signal and the data signals. An embedded FPGA core may be enabled to provide an interconnect to a chip. Additionally, software may enable a wide variety of features including bug fixes and product variations, all without changing the silicon.

Brief Summary Text (9):
Another aspect of the present invention concerns a method for programming a field programmable gate array (FPGA) comprising the steps of turning a system clock off, turning a scan clock on, forward shifting through a plurality of taps to a selected tap and serially programming one or more first registers from the selected tap.

Brief Summary Text (10):
The objects, features and advantages of the present invention include providing a method and/or architecture that may (i) add value above simple integration of an FPGA core versus an off-chip FPGA implementation, (ii) allow accessibility to internal signal points, (iii) enable one or more sections of a chip to be accessible to the FPGA core, (iv) allow a scan to minimize the hardware needed to implement an FPGA interconnect, (v) allow a backward scan shift function, (vi) allow a multiplexor to select a data feed from the FPGA, (vii) allow internal FPGA functions to control registers, (viii) implement multiplexors to minimize the hardware needed to implement the FPGA interconnect, (ix) design scales to FPGA size, SOC size and FPGA interconnect fragmentation needs, (x) allow the scan to scan other SOC chips for useful FPGA core inputs, and/or (xi) allow a multiplexor to scan taps, essentially creating a significant amount of inputs for the FPGA functions for a very low hardware cost.

Drawing Description Text (3):
FIG. 1 is a block diagram of an embedded FPGA core;

Detailed Description Text (2):
Referring to FIG. 1, a block diagram of a circuit 100 illustrating an embedded FPGA core is shown. The circuit 100 may be implemented, in one example, as a system on a chip (e.g., SOC). The circuit 100 generally comprises a circuit 102 and a memory 104. The circuit 102 may be implemented, in one example, as a field programmable gate array (e.g., FPGA). The FPGA 102 generally comprises a plurality of circuits 106a-106n and a core circuit 108. Each of the circuits 106a-106n may comprise one or more registers 110a-110n. The core circuit 108 may be implemented, in one example, as a FPGA core interconnect. A scalable architecture for an FPGA core

interconnect may be implemented with (i) a small FPGA core, (ii) a large FPGA core, (iii) a plurality of multiple independent cores and/or (iv) one or more clustered cores, independent of the size of the SOC. The circuit 102 may have an input 112 that may receive a signal (e.g., SCAN_IN_B). The circuit 102 may have an output 114 that may generate a signal (e.g., SCAN_OUT_B) in response to the signal SCAN_IN_B. The core portion 108 may have an input 116 that may receive a signal (e.g., FPGA_DWNLD) from the memory 104. The circuit 102 may have an input 118 that may receive a signal (e.g., SCAN_IN_F). The circuit 102 may have an output 120 that may generate a signal (e.g., SCAN_OUT_F) in response to the signal SCAN_IN_F.

Detailed Description Text (4):
A typical FPGA core may be implemented as an embedded FPGA core that may be used to enable logic to be programmed after the silicon has been produced. Such an FPGA core may be used in areas of the SOC 100 that are likely to change. The FPGA core may also be used in areas of the SOC 100 that need different programming for known areas of product variation, bug fixes and/or in-field upgrades.

Detailed Description Text (6):
In certain applications, tester limitations for running a scan may be 25 MHz. Devices may be designed to handle speeds up to the system clock speed for a scan which, for certain applications, may be up to 400 MHz. However, in other applications, devices may typically only run at 27 MHz. With such slow speed devices (e.g., audio/video decoders) slower speed clocks may be desirable to reduce power consumption. Therefore, on a part that can run a scan at 400 MHz, a system clock may run at 27 MHz. With the present invention, the FPGA may take 12.times.27 MHz clocks to load inputs, process a function and load embedded registers with the output.

Detailed Description Text (7):
The circuit 100 illustrates a bi-directional scan technique. The bi-directional scan technique may implement the circuits 106a-106n, the circuit 108 and a plurality of logic circuits 122a-122n. The circuits 106a-106n may each comprise a plurality of registers 110a-110n. The circuit 106n may have (i) an input 202 that may receive a first data signal (e.g., DATA), (ii) an input 204 that may receive a second data signal (e.g., SCAN_DATA_Bn) and (iii) an input 206 that may receive a third data signal (e.g., SCAN_DATA_Fa). The circuit 106n may have (i) an input 208 that may receive a first input signal (e.g., DATA_MUX_IN), (ii) an input 210 that may receive a second input signal (e.g., SCAN_Bn) and (iii) an input 212 that may receive a third input signal (e.g., SCAN_Fn). The circuit 106n may have an input 214 that may receive a first clock signal (e.g., SCAN_CLK) and an input 216 that may receive a second clock signal (e.g., SYS_CLK). The circuit 106n may have an output 218 that may generate a signal (e.g., SCAN_DATA_Fn) in response to the signals DATA, SCAN_DATA_Bn, SCAN_DATA_Fa, DATA_MUX_IN, SCAN_Bn, SCAN_Fn, SCAN_CLK and SYS_CLK. The circuits 106a-106n may have similar implementation. The signals SCAN_DATA_Fa-SCAN_DATA_F(n-1) may be generated by the circuits 106a-106(n-1) similar to the way the signal SCAN_DATA_Fn is generated by the circuit 106n.

Detailed Description Text (8):
The circuit 108 generally comprises a plurality of registers 220a-220n and a process circuit 222. In one example, the registers 220a-220n may each be 64-bit registers. However, other types of registers may be implemented accordingly to meet the design criteria of a particular application. The circuit 108 may have an input 224 that may receive the signals SCAN_DATA_Fa-SCAN_DATA_Fn. In one example, the input 224 may be n-bits wide. The circuit 108 may have an output 226 that may generate a plurality of signals (e.g., D1-Dn). In one example, the output 226 may be n-bits wide. The signals D1-Dn may be generated in response to the signal FPGA_DWNLD and the signals SCAN_DATA_Fa-SCAN_DATA_Fn. The circuit 108 may have an output 228 that may generate a plurality of signals (e.g., C1-Cn). The output 228 may be, in one example, n-bits wide. The signals C1-Cn may be generated in response to the signal FPGA_DWNLD and the signals SCAN_DATA_Fa-SCAN_DATA_Fn. The circuit 108

may have an output 230 that may generate a control signal (e.g., SCAN_CLK_CNTL) in response to the signal FPGA_DWNLD.

Detailed Description Text (9):
The circuits 122a-122n generally each comprise a multiplexor 233. In one example, the multiplexor 233 may be implemented as a 2 to 1 multiplexor. However, other types of multiplexors may be implemented accordingly to meet the design criteria of a particular application. The circuit 122a may have an input 232 that may receive the signal SCAN_DATA_Fa, an input 234 that may receive the signal C1, and an input 236 that may receive the signal D1. The circuit 122a may have an output 238 that may generate the signal SCAN_DATA_Bb in response to the signals SCAN_DATA_Fa, C1 and D1. The circuits 122a-122n may have similar implementation. For example, the signals SCAN_DATA_Bc-SCAN_DATA_Bn may be generated by the circuits 122b-122n similar to the way the signal SCAN_DATA_Bb is generated by the circuit 122a.

Detailed Description Text (11):
A method for programming the FPGA core 108 may comprise one or more of the following steps (i) turning the signal SYS_CLK off, (ii) turning the signal SCAN_CLK on, (iii) forward shifting through a plurality of taps to a selected tap and/or (iv) serially loading the FPGA registers from the selected tap.

Detailed Description Text (12):
Scanning may be used to minimize the hardware needed to implement the FPGA interconnect. Adding (i) a backward scan shift function, (ii) a multiplexor to select the data feed from the FPGA, and (iii) internal FPGA functions for control of what particular registers to load from the scan in and what registers to load FPGA process data to on the scan out may enable such a hardware implementation. Using the scan approach, even other SOC chips may be scanned for useful FPGA core inputs, using spare SOC I/Os or using a scan link to the chip, with taps on the input.

Detailed Description Text (14):
The logic gate 302 may have an input 310 that may receive an input signal (e.g., DATA_MUX_IN), an input 312 that may receive an input signal (e.g., SCAN_Bn) and an input 314 that may receive an input signal (e.g., SCAN_Fn). The logic gate 302 may have an output 316 that may generate a signal (e.g., OR2) in response to the signals DATA_MUX_IN, SCAN_Bn, and SCAN_Fn.

Detailed Description Text (15):
The multiplexor 304 may have an input 318, an input 320, an input 322, an input 324 and an output 326. The input 318 may pass through the signal DATA to the output 326 when the signal DATA_MUX_IN is active. The input 320 may pass through the signal SCAN_DATA_Bn to the output 326 when the signal SCAN_Bn is active. The input 322 may pass through the signal SCAN_DATA_Fa to the output 326 when the signal SCAN_Fn is active. The input 324 may receive the signal OR2 from the output 316. The multiplexor 304 may have an output 326 that may generate a signal (e.g., MUX4) in response to one or more of (i) the signal DATA, (ii) the signal SCAN_DATA_Bn, (iii) the signal SCAN_DATA_Fa and/or (iv) the signal presented at the output 330.

Detailed Description Text (16):
The logic gate 306 may have an output 328 that may generate a signal (e.g., OR6) in response to the signals SCAN_CLK and SYS_CLK. The flip flop 308 may have an output 330 that may generate the signal SCAN_DATA_Fn in response to the signals MUX4 and OR6. The circuits 110a-110n have a similar implementation. For example, output signals may be generated by the circuits 110b-110n similar to the way the signal SCAN_DATA_Fn is generated by the circuit 110a.

Detailed Description Text (18):
The multiplexors 402a-402n may each receive eight inputs. The multiplexors 402a-402n may have an input 414 that may receive a signal (e.g., CNTRL2) which may be 3-

bits wide signal. The multiplexors 402a-402n may have a plurality of outputs 416a-416n that may generate a plurality of signals (e.g., FPGA_INO-FPGA_INn-1). The signals FPGA_INO-FPGA_IN(n-1) may be generated in response to the eight inputs presented to each of the multiplexors 402a-402n. Instead of using a scan, the multiplexors 400a-400n may be used to load any 8 inputs from 64 register bits, in 1 to 8 cycles of the signal SCAN_CLK. If mapped to the second level of multiplexors 402a-402n, any of the 64 bits may be the 8 inputs to the FPGA 108 '.

Detailed Description Text (19):
The circuit 404a generally comprises a logic gate 302', a multiplexor 304', a logic gate 306' and a flip flop 308 '. In one example, the logic gate 302' may be a two input NOR gate, the multiplexor 304' may be a 3 to 1 multiplexor and the logic gate 306' may be a two input OR gate. The flip flop 308' may, in one example, be a D-type flip flop. However, other types of logic gates, multiplexors and/or flip flops may be implemented accordingly to meet the design criteria of a particular application. The logic gate 302' may have an output 316' that may generate a signal OR2' in response to a control signal (e.g., CNTRL3) and a control signal (e.g., SCAN_MODE). The multiplexor 304' may have an output 326' that may generate the signal MUX4' in response to (i) the signal FPGA_IN01, (ii) a data signal (e.g., SCAN_DATA) and (iii) the signal OR2'. The logic gate 306' may have an output 328' that may generate a signal OR6' in response to the signals SCAN_CLK and SYS_CLK. The flip flop 308' may have an output 418a that may generate the signal FPGA_IND01 in response to the signals MUX4' and OR6'. The circuits 404b-404n may be similar to the circuit 404a. Therefore, the output signals FPGA_IN02-FPGA_IN0n may be generated by the circuits 404b-404n similar to the way the signal FPGA_IN01 is generated by the circuit 404a.

Detailed Description Text (21):
The circuit 406 may have an output 434 that may generate a signal (e.g., IN1), which may be 64 bits wide, in response to the signals COL and ROW. The circuits 408a-408n may implement components similar to the components of the circuits 106a-106n. The circuit 408a may have an output 436 that may generate a signal OUT1 in response to the signals IN1, DATA_MUX_IN, SCAN_MODE, FPGA_DATA, DATA, SCAN_DATA, SYS_CLK and SCAN_CLK. The circuits 408b-408n may be similar to the circuit 408a. For example, the output signals of the circuits 404b-404n may be generated similarly to the way the signal OUT1 is generated by the circuit 404a.

Detailed Description Text (23):
Referring to FIG. 5, a circuit 100" is shown implementing another alternate embodiment of the present invention. The circuit 100" may implement a combination scanning/multiplexing technique. The circuit 100" may comprise a number of flip-flops 550a-550n, a number of taps 552a-552n, a number of multiplexors 554a-554n, a number of multiplexors 556a-556n. Each of the flip-flops 550a-550n may comprise, in one example, eight flip-flops. Each of the flip-flops 550a-550n may have one of tap 552a-552n. The first level of multiplexors 554a-554n and the second level of multiplexors 556a-556n may be implemented to minimize the interconnections of the circuit 100". In one example, both the first and second level of multiplexors may be 8 to 1 multiplexors. However, other types of multiplexors may be implemented accordingly to meet the design criteria of a particular application. Using the MUX approach to scan taps, up to 64 of the 512 bits may be accessible to the FPGA as inputs in eight cycles. Such an implementation provides a significant amount of inputs for a very low hardware cost (sixteen 8 to 1 multiplexors in the FIG. 4 example). The implementations illustrated in FIG. 2, FIG. 3 and FIG. 4 may permit use and template programming.

Detailed Description Text (24):
Design scales may be implemented to accommodate FPGA size, SOC size, and FPGA interconnect fragmentation needs. If part of the FPGA has access to chip I/O, part of the FPGA may use the scan approach and part of the FPGA may use the mux approach, with any combination of the approaches available.

Detailed Description Text (26):
The present invention may enable accessibility by the FPGA core to embedded logic and registers elsewhere on the SOC. Such an implementation may add to the value of using the FPGA core to fix bugs after the SOC is in silicon. The time and cost of new revisions of silicon is saved. The usefulness of the embedded FPGA core may be expanded to permit access to most of the registers in the SOC for use in processing. Future enhancements and upgrades to the SOC can use methods of the present invention, providing earlier prototypes and longer life parts. Earlier prototypes, longer product life, customer specific upgrades, bug fixes in development and bug fixes in the field all translate to development cost savings and in product competitive sustainability.

CLAIMS:

1. A method for programming a field programmable gate array (FPGA) and implementing an FPGA interconnect to enable a wide range of system on a chip (SOC) register access for FPGA inputs and outputs to update one or more first registers comprising the steps of: (A) turning a system clock off; (B) turning a scan clock on; (C) forward shifting through a plurality of taps to a selected tap; and (D) serially programming said one or more first registers from the selected tap to establish an FPGA function.

5. The method according to claim 1, further comprising the steps of: turning said scan clock off; executing an FPGA process; and generating outputs to be scanned into one or more of said first registers.

6. The method according to claim 5, further comprising the steps of: turning said scan clock on; backward shifting to a starting point with one or more new FPGA generated register values embedded where appropriate; and turning said system clock on.

9. The method according to claim 1, further comprising the step of loading any of one or more bits of said first registers in one or more cycles of said scan clock.

10. The method according to claim 1, further comprising the step of looping an end of a scan chain to an input of said FPGA.

11. The method according to claim 1, wherein any part of said FPGA that implements a chip I/O implements a scan approach, a multiplexor approach or any combination of said scan approach and said multiplexor approach.

12. The method according to claim 11, wherein said scan approach allows said FPGA to use inputs of other SOC chips.

15. The apparatus according to claim 13, wherein said second clock signal comprises a scan clock signal.

16. The apparatus according to claim 13, wherein said first control signal comprises a scan clock control signal.

26. A method for programming a field programmable gate array (FPGA) comprising the steps of: downloading a FPGA program; turning a system clock off; turning a scan clock on; scanning a plurality of registers; and programming said plurality of registers with an FPGA output having the FPGA program in response to said scan clock.

Previous Doc Next Doc Go to Doc#